2

AD-A278 195

RL-TR-94-9
Final Technical Report
March 1994

PHOTONIC FAST PACKET SWITCHING USING OPTICALLY PROCESSED CONTROL

Princeton University

Paul R. Prucnal

94-11372

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

94 4 14 016

RL-TR-94-9 has been reviewed and is approved for publication.

APPROVED: [illegible]

JOHN L. STACY
Project Engineer

FOR THE COMMANDER: [illegible]

LUKE L. LUCAS, Colonel, USAF
Deputy Director of Surveillance & Photonics

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE March 1994	3. REPORT TYPE AND DATES COVERED Final Jul 92 - Jul 93

4. TITLE AND SUBTITLE PHOTONIC FAST PACKET SWITCHING USING OPTICALLY PROCESSED CONTROL	5. FUNDING NUMBERS C - F30602-92-C-0040 PE - 62702F PR - 4600 TA - P2 WU - PK
6. AUTHOR(S) Paul R. Prucnal	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Princeton University Dept of Electrical Engineering Princeton NJ 08544	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Laboratory (OCPA) 25 Electronic Pky Griffiss AFB NY 13441-4515	10. SPONSORING/MONITORING AGENCY REPORT NUMBER RL-TR-94-9

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: John L. Stacy/OCPA/(315) 330-2937

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

In this research program, we have studied architectural and performance modeling issues which are central to the development of high-speed optical networks. We have focussed primarily on the architecture of a broadband photonic packet switch. The use of optical processing to perform switch routing functions permits real-time routing of packets at high speed. The architecture of a 2x2 photonic packet switching node using optically-processed fixed-directory routing, contention resolution (using deflection routing), and synchronization was investigated, and is presented in detail in this final report. Deflection routing is well-suited for contention resolution using processing technologies in which buffers are expensive or not available, as is presently the case with optical technology. Simplified optically-processed self-routing procedures are found for banyan (baseline, shuffle-exchange and crossover) and lattice networks. Although lattice networks require a larger number of switching elements than banyan networks, they have a simpler interconnection field, and unlike banyan networks, the self-routing rule for lattices can avoid any internal blocking.

14. SUBJECT TERMS Communications, Optics, Networks	15. NUMBER OF PAGES 36
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

1. Introduction

In future fiber-optic broadband integrated digital services networks, the large data rates in the fiber-optic transmission links as well as the large number of packets transmitted per second will place severe demands on the required transmission bandwidth and reconfiguration speed of packet switches. Although electronic switching technology has already achieved high switching speeds, it is quite possible that it will be difficult for electronic switches to match the transmission bandwidths that the fiber-optic links can provide. Thus photonic switches may be needed to provide a transmission bandwidth that matches the aggregate bandwidth of the services carried by the fiber. If packet switching is used, then the reconfiguration speed of the photonic switch must also be fast.

The reconfiguration speed of the photonic switch is determined not only by the time required for the switch to change its state, but the processing time required to determine the appropriate state of the switch. Core-and-edge logical network structures[1] for packet switches distribute the processing burden of low-level functions, such as routing, within the core of the network, and perform high-level functions, such as session setup, at the periphery of the network. The high-level functions require a large amount of slow processing, and can be performed easily with electronics. The low-level functions, on the other hand, require relatively simple processing, but must be performed at high-speed and completed in a time interval t' less than the packet length T_p,

$$t' \leq T_p, \quad (1)$$

so that the switch is ready to route the next packet as soon as it arrives. These high-speed low-level functions include:

1. Routing of packets from input to output by reading the packet address header and setting the switch permutation; in general, packet length information must also be read;
2. Resolving contention of packets at multiple input ports for a single output port according to an established priority scheme, in a manner that minimizes packet loss (e.g., by storing packets in a buffer or misrouting packets); in some architectures such as the banyan, internal contention may occur within the switch as well;
3. Synchronization of packets at the switch input.

One proposed method of performing the low-level functions is to use an electronic overlay network[2]. As illustrated in Fig. 1, the header of each packet is stripped, photodetected, and sent to a self-routing electronic switch that is topologically identical to the photonic switch. In the electronic overlay network, the routing information contained in the packet header determines the state of each electronic switching element, which in turn determines the state of the homologous optical switching element. Once the electronic switch is completely set, the optical packet (including the header and the data packet) is sent through the photonic switch. To avoid a data flow bottleneck at the switch input, the time t' required to electronically process the routing information and set the switching elements, must be less than T_p. For example, if the data rate is 5 Gbit/sec the length of an ATM packet is T_p=85 ns. It may be difficult for electronic processing to satisfy Eq. 1 in some fast packet switching applications.

The processing speed of the low-level functions can be increased by using parallel processing or pipelining[3]. A K-fold increase in speed can be obtained by connecting a system of K processors in parallel, and sequentially allocating the input data to the individual processors. However, the hardware required to replicate the processing units K times and sequentially allocate the data to the K individual processors may be complex. A K-fold increase in speed can also be obtained by forming a pipeline which partitions the low-level processing functions into a sequence of K discrete processing stages, of duration t'_i, for i=1,...,K. In this case, processing of the low-level functions must be completed at each stage of the pipeline in a time interval less than the packet length, that is,

$$t'_i \leq T_p, \quad \forall\ i=1,\ldots,K. \qquad (2)$$

As bit rates increase, it will become increasingly difficult for electronic processing to satisfy Eq. 2, even if parallel processing or pipeling is used. This suggests turning to optical processing of low-level functions to take advantage of its speed, and in some cases, its parallelism.

2. Generalized optical routing controller architecture

Routing strategies that are based on simple algorithms are the most feasible to implement with optical processing, because the present capability of optical processing is rather limited. An example of a routing strategy requiring only minimal processing is "deterministic" routing, in which the routes for all source-destination pairs are specified in advance. As a special case, "fixed-directory" routing maintains an optical routing table at each switch containing an outgoing link for each destination address[4,5].

Fig. 2 provides an illustration of optical fixed-directory routing. The network shown consists of five switches, A through E. Seven stations, 1 through 7, are attached to the switches. Based on the destination address of the incoming packet, a decision is made at each switch as to which outgoing link the data should follow. For example, data arriving at switch A with destinations 3 or 4 are routed through switch B to switch D, to which stations 3 and 4 are attached. Data arriving at switch A with destinations 5 or 6 are routed through switch C to switch D, to which station 5 is attached, and to switch E, to which station 6 is attached. Optically-processed fixed-directory routing is most easily implemented for a fixed network topology, where the connections at each switch are set and left unchanged for long periods of time. The optical route register need only be altered when topological changes occur in the network (resulting, for example, from the failure of a switch or link). As an illustration, if the link between switches C and D fails in Fig. 2, switch C can then be reconfigured to route data for destination 5 through switch B.

The block diagram of a 2x2 photonic switching node using an optically-processed fixed-directory routing controller is shown in Fig. 3. This 2x2 node may, for example, be one element in a banyan or lattice interconnection network (as discussed in Sec. 3 below).

The packet layout is shown in the inset to Fig. 3. The packet is of duration T_p, and consists of a header followed by a data payload. The duration of a data bit is τ. The header is comprised of a framing pulse of duration τ followed by several identification fields. In Fig. 3, field #1 contains the packet destination address and field #2 contains the packet length. Other fields may also be contained in the header, but are not considered here.

The optical packet-synchronization processor aligns the incoming packets at both switch inputs with a local optical clock of period T_p. In this processor, the phase delay between the framing pulse and the local clock is measured by the framing-pulse recognition processor. A packet delay signal is generated which tunes the optical delay line, providing the required synchronization.

Part of the signal at each switch input is diverted to the corresponding optical routing-control processor, which identifies the packet length and address and determines the state of the switch. The propagation delay of the packet through the routing controller is matched by a passive optical delay at the switch input, to insure that the state control signal and the packet arrive simultaneously at the switch.

In the routing controller, the signal is directed to both the packet-length recognition processor and the address recognition processor. The address recognition processor reads field #1 in the packet header to determine the destination address of the packet. The routing look-up table determines whether the packet destination address corresponds to the bar state or the cross state of the switch (or neither if the packet has been misrouted). The state control signals from both routing controllers are compared in the state-conflict resolution processor, where any contention between the two state control signals is resolved. The winning state control signal is passed along to the gating pulse generator. The gating pulse generator also receives control signals from the packet-length recognition processors, which have read field #2 in each packet header and determined the duration of each packet. In this way, the gating pulse generator sets the state of the switch for a time duration corresponding to the longer of the two packets, and the packets are directed to the desired output ports. Depending on the switch technology used, the physical form of the gating pulse may be optical, electrical, magnetic or acoustic. If an opto-optic switch is used in concert with the optical packet-synchronization processor and optcial routing control processor previously described, then the entire switch fabric can be optical.

The detailed structure of the optical processors that perform the above-described functions, including recognition of the framing pulse, address and packet length, look-up of the address, resolution of state control conflicts, and tunable delay, will be presented below.

2.1 Generalized optical recognition of orthogonal field signatures

Each of the optical fields in the packet header contains a signal that is read by a field recognition processor. The signal r(t) in the mth optical field corresponds to an element $s_i(t)$ of a set of orthogonal optical signatures $\{s_{F(m-1)+1}(t), s_{F(m-1)+2}(t), \ldots, s_{F(m-1)+F_m}(t)\}$. For example, the 1st field contains an element $s_k(t)$ of a set of F_1 orthogonal address signatures, and the 2nd field contains an element $s_j(t)$ of a set of F_2 orthogonal packet-length signatures. The orthogonal optical signatures can be represented in the time-, code- or frequency-domain.

At the optical field recognition processor, the received signal r(t) is optically correlated with each address signature $s_i(t)$, $i=F_{m-1}+1, F_{m-1}+2, \ldots, F_{m-1}+F_m$. This is accomplished using a bank of optical correlators performing the operations $r(t)\bullet s_i(t)$, $i=F_{m-1}+1, F_{m-1}+2, \ldots, F_{m-1}+F_m$, as shown in Fig. 4. The output of this matched filter is a vector $\{r(t)\bullet s_{F(m-1)+1}(t), r(t)\bullet s_{F(m-1)+2}(t), \ldots, r(t)\bullet s_{F(m-1)+F_m}(t)\}$ where the entry $r(t)\bullet s_i(t)=1$ if $r(t)=s_i(t)$ and $r(t)\bullet s_i(t)=0$ otherwise. Thus the position of the unity entry in the vector indicates the signature of the optical field.

Optical field recognition using TD coding: address and packet-length

Though the orthogonal address signatures can be represented in the time-, code- or frequency-domains, the following discussion is restricted to the time-domain (TD) respresentation, without loss of generality. The packet structure with TD field signatures is illustrated in Fig. 5a. Here field #1 is represented by a time frame with slots numbered $1,2,\ldots,F_1$. The signature of the kth packet address corresponds to a "1" in the kth slot and "0's" in all other slots. Field #2 is represented by a time frame with slots numbered $F_1+1, F_1+2, \ldots, F_1+F_2$. The signature of the jth packet length corresponds to a "1" in the jth slot and "0's" in all other slots.

Referring to layout of the matched filter in Fig. 4, recognition of a field signature with a TD representation requires a set of time-orthogonal signatures $\{s_{F(m-1)+1}(t), s_{F(m-1)+2}(t), \ldots, s_{F(m-1)+F_m}(t)\}$. Assume that a local optical clock pulse $u(t)-u(t-\tau)$ is available with duration τ and period T_p (where u(t) is the unit step function), which defines the location of the 0th time slot. As shown in Fig. 5b, the ith time signature $s_i(t)=u(t-i\tau)-u(t-(i+1)\tau)$ is generated by passing the optical clock pulse through an optical delay of duration $i\tau$. Using the appropriate set of delays to generate the set of address signatures from the optical clock, an optical matched filter can be constructed to perform field recognition of the address and packet length, as follows.

For address recognition, the set of time-orthogonal address signatures $\{s_1(t), s_2(t), \ldots s_{F_1}(t)\}$ is generated using delays of duration $i\tau$, where $i=1,2,\ldots F_1$. If r(t) corresponds to the kth packet address signature, then the output of the matched filter is a vector with a "1" in the kth position and "0's" in all other positions. This vector is then fed into the address look-up table.

For packet-length recognition the set of time-orthogonal packet-length signatures $\{s_{F_1+1}(t), s_{F_1+2}(t), \ldots s_{F_1+F_2}(t)\}$ is generated using delays of duration $i\tau$, where $i=F_1+1, F_1+2, \ldots, F_1+F_2$. If $r(t)$ corresponds to the jth packet length signature, then the output of the matched filter is a vector with a "1" in the jth position and "0's" in all other positions. This vector is then fed into the gating pulse generator.

2.2 Optical routing look-up table

The output of the address recognition processor is an address vector, which is converted to the appropriate state control signal by the look-up table shown in Fig. 6. (As discussed below, the routing look-up table can be eliminated in the special cases where the switch has a banyan or lattice architecture.)

The input to the routing look-up table from the matched filter is a vector $\{r(t)\bullet s_1(t), r(t)\bullet s_2(t), \ldots, r(t)\bullet s_{F_1}(t)\}$ where the entry $r(t)\bullet s_k(t)=1$ if $r(t)=s_k(t)$ and $r(t)\bullet s_i(t)=0$ otherwise. In order to determine which destination addresses correspond to the bar or cross state, each element of the vector is fed through a toggle switch to an optical summer (a passive star coupler). The toggles are closed for the set of elements in the vector that represent addresses that correspond to the bar state. The toggles are open for the set of elements in the vector that represent addresses that correspond to the cross state, as well as other destination address for packets that have been misrouted to this switch. Note that the toggles can represent fixed connections which are permanently open or closed, or the toggles can be programmable, using, for example, electrooptic switches. In some cases the processor design can be simplified by locating the toggles ahead of the correlators in the matched filter.

With the optical look-up table, any address pulse that passes through a closed toggle will result in a logical "1" state control pulse, corresponding to the bar state. Any address pulse that encounters an open toggle will result in a logical "0" state control pulse, corresponding to the cross state.

2.3 Optically-processed deflection routing for state conflict resolution

The incoming packet addresses at both switch inputs result in state control signals, denoted "A" and "B", at the outputs of the respective routing look-up tables. Since the switch has no *a priori* information concerning the addresses of the incoming packets, the state control signals A and B may be in agreement or in conflict with one another.

When a conflict occurs, only one packet can be correctly routed, and the other packet must be blocked. The blocked packet can be handled in several ways: it can be dropped (and lost), stored in a buffer at (or prior to) the point of conflict, or misrouted. Losing packets is clearly undesirable because it reduces the throughput of the switch. A great deal of work has been done on switching architectures that attempt to minimize the number of lost packets by storing the blocked packet in a buffer at the crosspoint until the first opportunity to send it forward. In these architectures, flow control signals are used to establish when the packet can be moved forward. If conflicts occur faster than stored packets can be forwarded, then the buffer will eventually overflow and packets will be lost. To avoid internal conflicts, packets can be sorted at the input to the network into an order that corresponds to one of the allowed permutations of the switch[6]. For example, in the "Batcher-banyan" architecture[7,8], output conflicts (two packets destined for the same output port) can be avoided by removing the conflicting packet and placing it in a buffer or feeding it back to the input for retransmission at a later time (see Sec. 3.1 below for a more extensive discussion of banyan architectures). Another approach uses output buffering by providing multiple paths between input and output using multiple copies of the network. This reduces the load on individual copies of the network and increases the throughput[9,10].

Store-and-forward techniques are well-suited for electronic switches in which flow-control algorthims and buffers can be implemented inexpensively with electronic logic. If storage buffers are expensive or not available, as is presently the case with optical technology, then deflection or "hot-potato" routing can be used, provided the number of input links to a crosspoint is the same as the number of output links[11,12]. In deflection routing, internal conflicts are resolved by correctly routing one packet and misrouting the other. In this way, all packets are forwarded, without buffering and without loss. The priority of misrouted packets can then be elevated to avoid misrouting a packet indefinitely[13]. A great deal of analysis has already been performed to determine the throughput and delay penalties associated with misrouting packets in various network architectures. Maxemchuk has shown[14] that, compared to a Manhattan Street Network with an infinite number of buffers, 55-70% of the throughput can be obtained using delfection routing. The throughput of hot-potato routing has also been compared to store-and-forward routing for networks with symmetric connectivity diagrams, such as the recirculating perfect shuffle or rectangular grid[15]. It was shown[15] that the performance hot-potato routing decreases monotonically as the number of nodes increases; with several hundred nodes, the throughput of hot-potato routing is 30% of store-and-forward routing. This performance degradation can be compensated by increasing the link speed. However, it may be difficult to increase the link speed significantly if electronic overlay networks are used to implement the hot-potato routing control function.

As discussed above, deflection routing is well-suited for technologies in which buffers are expensive or not available, as is presently the case with optical technology. The state conflict resolution processor shown in Fig. 7 employs optically-processed deflection routing processing to resolve contention.

The inputs to the optical state confict resolution processor in Fig. 7 are two state control signals A and B, where an input 1 corresponds to the bar state, and an input 0 corresponds either to the cross state or to a packet which has been misrouted to this switch. The resolved state control is obtained by optically summing the inputs A and B with a passive optical coupler to obtain the output state control signal C. A non-zero value (≥ 1) of C corresponds to the bar state and a 0 corresponds to the cross state. With this scheme, the bar state has high priority, whereas the cross state and unrecognized addresses have low priority. If a conflict occurs, "low priority" packets are misrouted. As shown in the table in Fig. 7, if both A and B are 1, then C is greater than 1, and the bar state is set with no conflict. If both A and B are 0, then C is equal to 0, and the cross state is set with no conflict. However, if one control is a 0 and the other is a 1, then the packet destined for the bar state is correctly routed and the other is misrouted. A misrouted packet still contains its correct address, and might be correctly routed at the next stage, or again not recognized and misrouted. A misrouted packet may eventually reach the correct destination; otherwise it will arrive at an idle destination and can be retransmitted through the network. In this way, buffering is, in effect, performed by the propagation delay through the network and retransmission at the periphery.

2.4 Optically-processed packet synchronization

Packet synchronization is required to ensure that the 2x2 switch can simultaneously route packets at both inputs to both outputs. Furthermore, the recognition of time-orthogonal address signatures requires that the received signal be synchronized with the local signatures. Synchronization can be accomplished by matched-filter recognition of the framing pulse followed by a tunable delay, as described below.

2.4a Framing pulse recognition

For framing pulse recognition the set of time-orthogonal signatures $\{s_0(t),s_{1/r}(t),s_{2/r}(t),\ldots,s_1,s_{1+1/r}(t),s_{1+2/r}(t),\ldots,s_2,\ldots,s_L,s_{L+1/r}(t),s_{L+2/r}(t),\ldots,s_{L+(r-1)/r}(t)\}$ is generated at a matched filter. Here L+1 is the total number of time-slots in a maximum length packet and r is a positive integer (see Fig. 5a). The set of signatures is generated from an optical clock pulse of duration τ/r, using a set of $r(L+1)$ delays of duration $i\tau$, where $i=0,1/r,2/r,\ldots,1,1+1/r,1+2/r,\ldots,2,\ldots,L,L+1/r,L+2/r,\ldots,L+(r-1)/r$. The resolution of the framing pulse recognition is determined by $1/r$, which corresponds to a fraction of a time slot τ.

If the first (framing) pulse to enter the matched filter has the maximum overlap (correlation) with the pulse in the mth packet length signature, then the output of the matched filter is a vector with the maximum value in the mth position, smaller values in neighboring positions, and "0's" in all other positions. This vector then sets the tunable optical delay to $(L-m)\tau$. The framing pulse is then delayed to slot 0 (in phase with the optical clock) with accuracy τ/r, accomplishing the desired synchronization. Once the tunable optical delay is set, it must remain set (that is, inhibited from changing) for the duration of the packet. Since no *a priori* information about the packet length is available, it is assumed that the framing pulses are spaced no closer than the duration of the maximum length packet $(L+1)\tau$, and that the tunable optical delay is inhibited from changing during this time. After this time period, the control of the tunable optical delay is released, and the next packet can be synchronized with the clock.

2.4b Tunable optical delay

A tunable optical delay can be implemented with the variable-integer-delay line shown in Fig. 8. This design allows a large number of delays with fast reconfiguration time. To produce $M=r(L+1)$ possible delays, the feed-forward structure consists of $\log_2 M$ delay stages $k=1,\ldots,\log_2 M$ and an output stage. Thompson has shown analytically that a feed-forward structure requires fewer stages than a feed-back structure[16]. Each delay stage consists of a 2x2 optical switch, a connection to the next stage at one output, and a fixed optical delay in excess of the "connection" delay at the other output. The value of the fixed excess delay for the kth stage is $T_p/2^k$. Only one input is used to the first stage. The output stage consists of a 2x2 optical switch, where only one output is used. Each optical switch can be set in either the bar or the cross state. The state of a switch is set by the electrical control input, where a 0 at the control sets the 2x2 switch in the cross state, whereas a 1 sets the 2x2 switch in the bar state.

The state of the encoder is set by a control sequence $(c_1,c_2,\ldots,c_{\log M})$, where control bit c_k sets the state of the kth stage, and the output stage is set equal to 0 if the parity of the control sequence is odd, or to 1 if the parity of the control sequence is even. The output stage serves only to ensure that the delayed pulse always exits at the chosen output of the 2x2 optical switch. The control sequence for the jth slot is generated from the binary representation of the integer j, $(b_1, b_2, \ldots,b_{\log M})$, where b_1 is the most significant bit, according to the rule $c_1=\{$the complement of $b_1\}$, and for $i= 2,\ldots,\log_2 M$, $c_i= 0$ if $b_i=b_{i-1}$, otherwise $c_i=1$. Thus, control circuitry is required to convert the vector position output of the matched filter into the binary representation required to set the tunable delay. After the control sequence has set the encoder as described above, the sampled data will be delayed by an amount $j\tau$ in excess of the reference delays, accomplishing the desired time-division encoding operation.

An experimental demonstration of the variable-integer-delay line was previously reported[17] for 64 100 Mbit/sec channels, where T = 10 nsec and τ= 156.25 psec. The encoder required six delay stages $k=1,\ldots,6$ with time delays $D_k=10/2^k$ nsec, corresponding to fiber lengths $L_k=2.052/2^k$ m, where the index of refraction of the fiber core was $n_f=1.462$. To guarantee a time delay error of less than 10% of a time slot, then the aggregate positioning error must be less than

3.2 mm, or 0.53 mm per stage, which required careful trimming of the fiber lengths. In the demonstration, the maximum aggregate error in delay was measured to be 2.5% of a time slot. The reconfiguration time of the encoder was determined by the switching speed of the directional couplers. The maximum number of stages $\log_2 M$ that can be used is limited by the insertion loss. The insertion loss can be minimized in the integrated-optic delay stages by integrating all of the stages together, including the directional couplers and the waveguide delays, on a single substrate. The insertion loss in the fiber-optic delay stages can be compensated using optical amplifiers. In this way, the implementation of a variable optical delay capable of producing 1000 delays with subnanosecond reconfiguration time may be feasible.

2.4c Simplified synchronization scheme

The optical packet synchronization described in Sec. 2.4a is carried out using a framing pulse recognition processor of size $F=r(L+1)$, and a tunable optical delay, where the resolution of the framing pulse recognition is determined by $1/r$, which corresponds to a fraction of a time slot τ, and $L+1$ is the total number of time-slots in a maximum length packet. A simpler optical synchronization scheme requiring only $\log_2 r(L+1)$ framing pulse recognition processors of size $F=2$ is shown in Fig. 9.

The synchronization processor in Fig. 9 consists of $\log_2 r(L+1)$ stages. The input packet to the first stage arrives with an arbitrary phase. Since no *a priori* information is available about the packet length, it is assumed that the framing pulses are spaced no closer than the duration of the maximum length packet $(L+1)\tau$. The packet is directed to both the 1x2 photonic switch and the framing pulse recognizer. A passive delay is introduced before the 1x2 switch to match the propagation delay through the framing pulse recognizer and the delay of the gating pulse generator, so that the gating pulse and the packet arrive simultaneously at the switch.

At the j^{th} stage, the framing pulse recognizer determines whether the framing pulse is in the first or the second half of the interval $0 \le t \le T_p/2^{j-1}$, where $T_p=(L+1)\tau$. This is accomplished with an optical matched filter with $F=2$, as described in Sec. 2.1 and illustrated in Fig. 4. At stage #j, the set of time orthogonal signals $s_1(t)=u(t)-u(t-T_p/2^j)$ and $s_2(t)=u(t-T_p/2^j)-u(t-T_p/2^{j-1})$ are generated from a local optical clock pulse of duration τ/r and period T_p. To generate $s_1(t)$, the clock triggers an optical pulse of duration $T_p/2^j$; this pulse is delayed by $T_p/2^j$ to produce $s_2(t)$. The position of the local optical clock pulse defines the time origin, to which the framing pulse will be synchronized. If the framing pulse is located in the first half of the interval, then the vector output of the matched filter is (1,0); if the framing pulse is located in the second half of the interval, then the vector output of the matched filter is (0,1). For an input vector (0,1), the gating pulse generator sets the 1x2 switch in the bar state for a time interval T_p, and it is inhibited from changing during this time. For an input vector (1,0), the gating pulse generator sets the 1x2 switch in the cross state for a time interval T_p, and it is inhibited from changing during this time. The upper (bar) output of the switch enters a reference delay and the lower (cross) output enters a delay $T_p/2^j$. The two delays are summed and enter the next stage. After $\log_2 r(L+1)$ stages, the framing pulse is aligned with the local clock with an accuracy of τ/r.

3. Simplified self-routing in banyan and lattice networks

Optically-processed routing, synchronization and contention resolution have been discussed above for the generalized case of a 2x2 photonic switching node located in an arbitrary position in a network of arbitrary architecture. Routing takes place by identifying the final destination address in the packet header, and looking up the corresponding switch state in a fixed directory. This routing procedure can be simplified for several specific network architectures in which the position of each 2x2 switch within the network is used to determine the appropriate path to the destination. Simplifying the routing rule in this way allows the complexity of the optical routing processor to be reduced. Two cases will be considered in which a simplified routing rule can be used to reduce the complexity of the optical routing processor: banyan and lattice neworks. In fact the simplified routing rule for the banyan network can also be extented to the crossover, shuffle-exchange and baseline neworks, which have been shown to be isomorphic to one another[18].

3.1 Banyan networks: self-routing with one bit per stage

In the development of fast packet switching systems, extensive research has already been carried out on large switch architectures based on multistage interconnection networks (see Tobagi[19] for an excellent review of this subject). An advantage of some multistage architectures for packet switching is that the control need not be centralized, but may be distributed throughout the switch fabric. Global information about the addresses of all packets is not required to set the state of the switch in advance, as with centralized control procedures. This is particularly suitable for packet switching, where the switch has no *a priori* knowledge of the arrival time or destination address of a packet. Routing decisions can be made at each crosspoint, so that a packet can self-route through the multistage network. Another advantage of multistage architectures is that every path through the switch has the same transit time. A major disadvantage of some multistage architectures is that not all arangements of input--output connections can be made, resulting in blocked packets.

The construction of multistage architectures starts with four-port switching elements arranged in a binary tree (shown as the cross-hatched boxes and heavy lines in fig.10). In principle, an arbitrarily large 1xN binary tree could be constructed with only N-1 2x2 directional coupler switches (crosspoints) arranged in $\log_2 N$ stages (if N is a power of 2) interconnected by fiber amplifiers.

The distributed routing procedure in the binary tree is particularly simple (see fig.10). If the destination address of a packet is represented in binary form, then each bit of the address determines whether a stage should direct the packet to its upper (bit 0) or lower (bit 1) output, where the most significant bit controls the first stage, and each successive bit controls successive stages. Each stage need only examine one bit of the address. For example, in Fig.10 the address 001 indicates that the first stage should route the packet to the upper output, the second stage to the upper output and the third stage to the lower output. It is easily verified that using this procedure, the packet self-routes to output port 001. An NxN multistage switch constructed from distinct 1xN binary trees would consist of N binary trees for demultiplexing and N for multiplexing, requiring a total of about twice the number of crosspoints as a crossbar switch.

The required number of crosspoints in a multistage switch can be reduced to $(N/2)\log_2 N$ if the inputs share the binary trees. For example, the 8x8 banyan switch shown in Fig.10 requires only 12 crosspoints. Banyan architectures provide a single path from each input to each output, just as in the binary tree. The processing can be distributed among all the nodes, and packets can self-route through the switch by examining only one address bit at each stage. This self-routing capability makes the banyan network well-suited for fast-packet switching, where the state of the switch must change so quickly that centralized control may be difficult.

The reduction in the number of crosspoints achieved by the banyan architecture is accompanied by a major limitation: internal blocking can occur if two packets are destined for the same output of the same crosspoint. Such conflicts result in blocking one of the packets and correctly routing the other. Thus, not all input-output permutations are possible with the banyan switch, and the throughput of this architecture is severely limited. This is seen in Fig. 10, where two packets with destination addresses 111 and 110 need to simultaneously be routed to the lower output port of the second crosspoint in the second stage, resulting in an internal conflict.

As discussed above, when the switch has a banyan architecture, the switch output port address can simply be represented in binary form, rather than a time-orthogonal destination address signature as described in Sec. 2.1. Note that each bit in the binary representation is an element in an orthogonal set of optical signatures represented in time, frequency, wavelength, etc. Representing the address in binary form substantially reduces the size of field #1, from F_1 to $\log_2 F_1$. At the kth stage in the banyan switch the matched filter in Fig. 4 is reduced from a bank of F_1 correlators to a single correlator $r(t) \cdot s_k(t)$; in the case of time-orthogonal signatures, the signature $s_k(t)$ is derived from the local optical clock with a single optical delay of duration $k\tau$ as illustrated at the bottom of Fig. 10.

In accordance with the usual banyan routing procedure, if the kth bit of the binary address code is a 0 (1), then the output of the correlator is a 0 (1), and the switch should direct the packet to its upper (lower) output. To accomplish this in the upper routing processor, the scalar output of the correlator is fed directly into the state conflict resolution processor. In the lower routing processor, the scalar output of the correlator is logically inverted and then fed into the state conflict resolution processor. Inverting the lower correlator output of 0 (1) results in the cross (bar) state, which directs the packet to the upper (lower) output, as desired.

No address look-up table is needed with the banyan architecture, because the binary address corresponds to the switch output port, rather that the final packet destination address in the network. Thus, in the case of a banyan architecture, encoding the address in binary form substantially reduces the complexity of the address recognition processor and eliminates the need for an address look-up table, but requires an additional logical invert function in the lower processor.

The packet-length can also be encoded in binary form, whether or not the switch has a banyan architecture. Representing the packet-length in binary form substantially reduces the size of field #2, from F_2 to $\log_2 F_2$. The matched filter in Fig. 4 is reduced in size to a bank of $\log_2 F_2$ correlators. The output of the matched filter is a vector corresponding to the binary-coded packet-length, which is fed into the gating pulse generator. In general, encoding the packet-length in binary form substantially reduces the complexity of the packet-length recognition processor, and feeds a binary-encoded vector to the gating pulse generator, rather than a vector with a single-nonzero bit.

3.2 Lattice networks: self-routing by address ordering

In the case where the switch has an NxN lattice architecture, the optical look-up table can also be eliminated. The lattice network has a larger number of switching elements ($N(N-1)/2$) than the banyan ($(N/2)\log_2 N$), but has a simpler interconnection pattern without crossovers, and, unlike the banyan, can be arranged to avoid any internal blocking.

Say the output ports of the switch are numbered from top to bottom as 1 through N, as shown in the lattice network in Fig. 11 (for the case where N=8). Assume that part of the signal at each of the node input ports is diverted to an address recognition processor which reads field #1 in the packet header to determine the destination address of the packet, denoted as "T" for the upper input and "I" for the lower input. Then the routing rule at each switching node simply requires that the magnitude of the addresses be compared; the smaller address is routed to the upper output and the larger address to the lower output. Thus the routing rule in Fig. 11 is simple: if $T<I$ then select the bar state; if $T>I$ then select the cross state. If all destination addresses 1,...,N are input to the switch, then all packets are self-routed from input to output without internal blocking. This is in contrast to the case of the Banyan (baseline, crossover, and shuffle-exchange) switches, which can have internal blocking. If two or more destination addresses are equal (corresponding to output contention), then the destination addresses ranging in value between the duplicated address and the unused address will be misrouted, and must be retransmitted through the network. If one or more of the destination addresses are unused, then some node input ports will be idle; then T or I at the idle input port may assume a default value equal to the row number of that port.

A simple time-domain approach to implementing the address comparator optically is as follows. Referring to Fig. 11, consider a set of time-orthogonal optical signatures, where the synchronization pulses have been removed and the synchronized packets are separated by a guard band of duration equal to the packet header. Then the address pulse that arrives at one of the two switch inputs first represents the smaller address. If the address at the upper input arrives first, it triggers control input T to generate a gating pulse which sets the switch in the bar state, and the packet passes to the upper output. If the address at the lower input arrives first, it triggers control input I to inhibit the gating pulse generator, which sets the switch in the cross state, and the packet passes to the upper output. The duration of the gating or inhibit pulse is equal to a packet length, and the guard band following the packet insures that the gating pulse does not interfere with the next packet.

4. Self-routing in 2D lattice switching architectures

Some photonic switch technologies (for example, smart pixels), can be fabricated readily in two-dimensional (2D) planar arrays. 2D switching architectures, as shown in Fig. 12, which consist of a series of planar arrays of switching elements linked by optical interconnection fields, have received a great deal of attention in the literature.

A 2D architecture is easily derived from an NxN one-dimensional (1D) architecture by partitioning the N rows into K sets, and folding it accordion-style while preserving the interconnection field to form a KxN/K planar crossection. This construction was used to produce the omega-shuffle-exchange and crossover networks[18]. A 2D architecture constructed in this manner is equivalent to the corresponding 1D architecutre in every way except in the physical placement of the switches. Thus the self-routing architectures discussed for the 1D architectures above, such as the banyan and lattice, apply identically to the 2D architectures as well.

If smart pixel arrays are used to implement the switching arrays, only minimal processing logic may be available on each pixel, and it may be necessary that the self-routing rule be as simple as possible. In addition, local data storage on a smart pixel may be minimal or nonexistent. Therefore, architectures such as the lattice which avoid internal blocking and any requirement for packet buffers have an important advantage over architectures such as the banyan which can have internal blocking. For the same reason, link blocking should be handled by deflection routing.

The layout of a self-routing 2D lattice using smart pixels, formed by folding a 64x64 1D lattice with K=8 is shown in Fig. 12. As described above, each plane is formed by taking one column of the 1D lattice, dividing it into K=8 sets, and folding it accordion-style to form an 8x8 planar array. In this way the 1D column is laid out in a serpentine fashion on the surface of the plane as shown by the numbered inputs in Fig. 12. In Fig. 12, each rectangle in the planar array represents a self-routing 2x2 switching element with inputs on the front surface of the plane and outputs on the back surface. The 2x2 switching elements in Fig. 12 are functionally equivalent to the 2X2 switches with trigger and inhibit gating controls shown in Fig. 11. Each 2x2 switching element may contain more than one smart pixel, as, for example, would be the case in a 2D 2x2 vector-matrix multiplier. The simple interconnection field of the 2D lattice in Fig. 12 (all connections are simply straight ahead) provides for easy alignment and stacking of the smart pixel array planes. For collimated input beams, no additional optics are required to provide the interconnections. Note that in the second plane the pairs of inputs which are switched are shifted by one position relative to the first plane, as is the case in the first and second columns of the 1D lattice in Fig. 11. Because the layout of the pixels in the plane repeats after two stages, the entire switch can be constucted simply with only the first two stages followed by feedback to the input. Although this reduces the complexity of the switch, additional latency is introduced into the switching processs, which then must be matched by a like delay between succesive packets, which in turn reduces the throughput of the system. As described in Sec. 3.2 for the 1D case, self-routing can be accomplished by simple time-domain encoding, such that the first packet to arrive triggers a gating or inhibit signal. In this way, no additional logic is required on the 2x2 switching element to perform routing.

5. Implementations of optically-processed fixed-directory routing and synchronization

The optically processed routing controller and synchronization described in above and illustrated in Figs. 3-12, can be implemented with only passive optical components (fiber-optic delays, summers and splitters), a low-duty-cycle optical clock (e.g., a mode-locked laser), a non-paralyzable gating pulse generator (in which a fast trigger pulse generates a gating pulse of duration T_p), and fast optical correlators. All of these components are available "off-the-shelf" except for the fast optical correlators.

As discussed above (see Eqs. 1 and 2), the control functions of a photonic fast packet switch, including routing, contention resolution, and synchronization, must be completed in a time interval less than the packet length, so that the switch is ready to route the next packet as soon as it arrives. This is the motivation for considering optical implementations of the low-level functions, and, indeed, for choosing a routing strategy such as fixed-directory which minimizes the complexity of these functions. Clearly the passive fiber-optic components impose no bandwidth limitations on the system, as compared to the bandwidth of the transmission channel. That is, the latency due to propagation of the packet header through the passive optical processors does not impose any processing delay or a bandwidth limitation on the switch; this latency only needs to be matched by an equal passive delay at the switch input, to guarantee the synchronous arrival of the packet and the gating pulse. The local optical clock can consist of a mode-locked semiconductor laser, with period equal to the maximum packet length and pulse-width equal to the bit duration, and also imposes no bandwidth limitations on the system. The type of the gating-pulse generator used depends on the nature of the control pulse required for the photonic switch; for example, an electrooptic switch requires an electrical gating pulse. Although fast-triggered pulse generators and driving amplifiers do not have bandwidths that match the transmission channel, they should not significantly constrain the bandwidth of the system. Therefore, asided from passive propagation delay, the total processing speed of the optical synchronization and routing controllers is determined by the speed of a single gate which performs the thresholding and pulse generation functions. As described below, photoconductive logic gates suited to this task have been demonstrated with response times that approach 10 ps; all-optical soliton logic gates have been demonstrated with reponse times of 300 fs. The speed of the electronic processing required to perform the tasks of synchronization, address recognition, routing look-up, and state conflict resolution, using current technology, would be well in excess of 1 ns. Therefore, with a sufficiently fast optical correlator (satisfying Eqs. 1 and 2), then the optically-processed routing control and synchronization can be performed in real-time, for <u>arbitrarily high bit-rates</u>.

We have previously performed numerous experiments using optical correlation to carry out optically-processed routing control of photonic switches. Some of these experimental demonstrations of optically-processed self-routing are summarized in Ref.20. These experiments include packet-address recognition and look-up using: optical code-division routing at 100 Mchip/s with 32-chip packet headers, and a lithium niobate electrooptic Mach-Zehnder integrated-optic modulator as the photonic switch[21-23]; optical code-division routing at 12.5 Gchips/s with 125-bit packet headers, and an integrated-optic 8x8 crossbar photonic switch[24]; optical time-division routing[25-27] at 12.5 Gbits/s with 125-bit packet headers, and a lithium niobate electrooptic Mach-Zehnder integrated-optic modulator as the photonic switch[28]; optical self-clocked pulse-interval routing at 12.5 Gbits/s with 125-bit packet headers, and a lithium niobate electrooptic Mach-Zehnder integrated-optic modulator as the photonic switch[29]; optical binary routing at 10 Gbits/s with a 3-bit packet header and a 4x4 banyan-type photonic switch[30].

In most of the above experiments[21-24,26-29], fast correlation is performed by passively delaying and summing the optical address signature, followed by photodetection and electronic thresholding. The detection and thresholding process limited the speed of the processing. A faster optoelectronic correlation technique was introduced for binary self-routing through a 4x4 banyan-type photonic switch[30] using photoconductive "AND" gate[31].

In order to perform the correlation operation with a speed that matches the bandwidth of the transmission channel, all-optical correlation can be used, in which the position (in time or space) of a pulse is controlled by another ultrashort optical pulse. Soliton switching has been experimentally shown to provide very-high speed, very-high efficiency all-optical correlation. The basic motivation for soliton switching lies in the fact that, unlike non-soliton optical pulses, fundamental solitons have a constant optical phase across their envelope. This feature is due to the interplay between self-phase modulation and group-velocity dispersion that leads to soliton formation in fibers. This constant phase allows solitons to behave as single entities in phase- sensitive switching, thus avoiding pulse breakup and considerably improving the switching efficiency and the cascadability of soliton switches. In all-optical soliton switching in birefringent fibers, the phase shift induced to a pulse propagating along a principal axis of the fiber is provided by another orthogonally-polarized soliton, through a soliton interaction. In such an interaction, the two pulses, initially having different group velocities, shift their central frequencies in opposite directions in a way that after interaction they move together[32]. This is due to the cross-phase modulation between the orthogonal modes that can compensate for the effects of the linear birefringence. Hence this "soliton dragging" allows for a time shift of the solitons that can be used for time switching if the output pulses are probed in a time window at the order of the pulse width. Low-energy ultrafast logic gates based on soliton dragging interactions have been demonstrated[33,34], where bit-rates at the order of 200 Gbps can be achieved. The switching energies of such gates were brought down to 5.8 pJ[33], and a switching energy of about 1 pJ seems to be realizable[35]. An optical AND operation that can be used for ultrafast time sampling of multiplexed optical pulses and optical correlation can be realized by using a soliton trapping interaction[32,33]. By carefully choosing the parameters of the birefringent fibers and the pulse amplitudes, the two orthogonally polarized pulses propagating along the fiber give at the output a spectrum that has two lobes surrounding the position of the spectrum of one of the pulses in the absence of the other[33] Therefore, a narrow spectral filter (such as a Fabry-Perot etalon) at the output that would be centered on one of the lobes of the bifurcated spectrum would provide the optical AND operation.

6. Conclusion

The use of optical processing of high-speed low-level functions permits routing functions to be completed in a time less than a packet length, so that the switch is ready to route the next packet as soon as it arrives. Such functions include real-time routing, contention resolution and synchronization. Routing strategies that are based on simple algorithms are the most feasible to implement with optical processing, because the present capability of optical processing is rather limited. An example of a routing strategy requiring only minimal processing is "deterministic" routing, in which the routes for all source-destination pairs are specified in advance. As a special case, "fixed-directory" routing maintains a routing table at each switch containing an outgoing link for each destination address. Such a routing table is easily implemented with optics.

The structure of a 2x2 photonic packet switching node using optically-processed fixed-directory routing, contention resolution and synchronization is presented. Recognition of orthogonal (in the time, code or frequency domain) optical signatures, carrying address, packet length and framing pulse information, can be performed with an optical matched filter. The output of the address recognition processor is an address vector, which is converted to the appropriate state control signal by an optical look-up table. The optical look-up table consists of toggles which are closed for the set of elements in the vector that represent addresses that correspond to the bar state, and open for the set of elements in the vector that represent addresses that correspond to the cross state. The toggles can represent fixed connections which are permanently open or closed, or the toggles can be programmable, using, for example, electrooptic switches.

The incoming packet addresses at both switch inputs result in state control signals at the outputs of the respective routing look-up tables. Since the switch has no *a priori* information concerning the addresses of the incoming packets, the state control signals may be in agreement or in conflict with one another. The optical state conflict resolution processor produces an output state control signal that resolves contention using deflection routing. Deflection routing is well-suited for technologies in which buffers are expensive or not available, as is presently the case with optical technology.

Packet synchronization is required to ensure that the 2x2 switch can simultaneously route packets at both inputs to both outputs. Synchronization can be accomplished by matched-filter recognition of the framing pulse followed by a tunable delay. A tunable optical delay can be implemented with a variable-integer-delay line. This design allows a large number of delays with fast reconfiguration time.

Simplified self-routing procedures can be found for specific network architectures in which the position of each 2x2 switch within the network determines the appropriate path to the destination. For banyan (baseline, shuffle-exchange and crossover) networks, routing can be accomplished optically with one bit per stage and no optical look-up table. Banyan networks have the disadvantage of internal blocking. For lattice networks the optical look-up table can also be eliminated, and routing can be accomplished by simply ordering the magnitudes of the two destination addresses at each 2x2 switch. Although lattice networks require a larger number of switching elements than banyan networks, they can be arranged to avoid internal blocking (by using the "ordering" routing rule described above). A simple time-domain approach for optical implementation of the "ordering" rule is described, such that the first packet to arrive triggers a gating or inhibit signal. In this way, no additional logic is required on the 2x2 switching element to perform routing.

2D switching architectures are attractive for switch technologies which are readily fabricated in arrays, such as smart pixels. A 2D architecture is easily derived from a 1D architecture (such as the banyan or lattice) by folding each column in the 1D architecture in a serpentine fashion, while preserving the interconnection field, to form each plane of the 2D switch. In this way, the optically-processed self-routing architectures described for the 1D architecture applies identically to the 2D architecture. The layout of a self-routing 2D lattice, using 2x2 switches comprised of smart pixels, is described. Based on the "first-arrival" ordering rule, no additional logic is required to perform routing.

The optically processed routing controller and synchronization processing can be implemented with only passive optical components (fiber-optic delays, summers and splitters), a low-duty-cycle optical clock (e.g., a mode-locked laser), a non-paralyzable gating pulse generator, and fast optical correlators. *All of these components are available* "off-the-shelf" except for the fast optical correlators. A fast optoelectronic correlation technique was introduced for binary self-routing through a 4x4 banyan-type photonic switch using photoconductive "AND" gate. In order to perform the correlation operation with a speed that matches the bandwidth of the transmission channel, all-optical correlation should be used. If a sufficiently fast optical correlation (satisfying Eqs. 1 and 2) can be implemented in the future, then the optically-processed routing control and synchronization for packet switching can be performed in real-time.

References

1 P. Cinato and A. de Bosio, "Optical technology applications to fast packet switching," in Photonic Switching, J.E. Midwinter and H.S. Hinton, ed., OSA Proceedings 3. pp. 233-236, 1989

2 A. deBosio, C. DeBernardi and F. Melindo, "Deterministic and statistic circuit assignment architectures for optical switching systems," Topical Meeting on Photonic Switching, Tech. Dig. Series, Vol. 13, paper ThB2, pp. 35-37, 1987

3 L.T. Wu, S.H. Lee and T.T. Lee, "Dynamic TDM--a packet approach to broadband networking," IEEE Int'l Conf. on Commun., Proc., Vol 3, paper No. 46,, pp. 1585-1592, 1987

4 M. Schwartz, Telecommunication Networks: Protocols, Modeling and Analysis, Addison-Wesley Publishing Co., Reading, MA, 1987

5 P.R. Bell and K. Jabbour, "Review of point-to-point network routing algorithms," IEEE Communications Magazine, Vol. 24, No. 1, pp. 34-38, 1986

6 K.E. Batcher, "Sorting networks and their applications," in AFIPS Proc. 1968 Spring Joint Computer Conference, Vol. 32, pp.307-314

7 A. Huang and S. Knauer, "Starlite: a wideband digital switch," in Proc. Globecom '84, Atlanta, pp.121-125?

8 M.J. Narasimha, "The Batcher-banyan self-routing network: universality and simplifications," IEEE Trans. Commun., Vol. 36, No. 10, pp. 1175-1178, 1988

9 C.P. Kruskal and M. Snir, "The performance of multistage interconnection networks for multiprocessors," IEEE Trans. on Computers, Vol. C-32, No. 12, pp. 1091-1098, 1983

10 M. Kumar and J.R. Jump, "Performance of unbuffered shuffle-exchange networks, " IEEE Trans. on Computers, Vol. C-35, No. 6, pp. 573-577, 1986

11 P. Baran, "On distributed communications networks," IEEE Trans. on Comm. Sys., pp.1-9, March, 1964

12 N.F. Maxemchuck, "Regular and mesh topologies in local and metropolitan area network, " AT&T Tech. J., Vol. 64, No. 7, pp. 1659-1686, 1985

13 J.R. Sauer, "A multi-Gb/s optical interconnect," OE/Lase 1990, paper #22, Conf 1215, Digital Optical Computing 2

14 N.F. Maxemchuck, "Comparison of deflection and store-and-forward techniques in the Manhattan street and shuffle-exchange networks," IEEE Infocom 89, Ottawa, Ontaro, Canada, April 25-27, 1989, pp. 800-809

15 A.S. Acampora, Center for Telecommunications Research, Annual Tech. Rep., pp. 1-4, November 1989

16 R.A. Thompson, "Optimizing photonic variable-integer-delay circuits," Topical Meeting on Photonic Switching Tech. Dig., Vol. 13, Paper FD4, pp. 241-243, 1987

17 P.R. Prucnal, M.F. Krol, and J.L. Stacy, "Demonstration of a rapidly-tunable optical time-division multiple access coder," Photonics Tech. Lett., Vol. 3, No. 2, pp.170-172, 1991

18 T.J. Cloonan, "Topological equivalence of crossover networks and modified data manipulator networks," Appl. Opt., Vol. 28, p. 2494, 1989.

19 F.A. Tobagi, "Fast packet switch architectures for broadband integrated services digital networks," Proc. IEEE, Vol. 78, No. 1, pp. 133-167, 1990

20 P.R. Prucnal and P.A. Perrier, "Self-routing photonic switching with optically-processed control," Optical Engineering, Vol. 29, No. 3, 170-182, 1990

21 Prucnal, P.R., Santoro, M.A. and Fan, T.R. (1986). "Spread spectrum fiber [illegible]tic local area network using optical processing," *IEEE J. Lightwave Tech.*, LT4 #5, 547-554.

22 Santoro, M.A. and Prucnal, P.R. (1987). "Asynchronous fiber optic LAN using CDMA and optical correlation," *Proc. IEEE*, 75 #9, 1336-1338.

23 P.R. Prucnal, D.J. Blumenthal, and P.A. Perrier, "Photonic switch with optically self-routed bit switching," IEEE Communications Magazine, Vol. 25, No. 5, pp. 50-55, 1987

24 D.J. Blumenthal, P.R. Prucnal, L. Thylen, and P. Granestrand, "Performance of an 8x8 $LiNbO_3$ switch matrix as a gigahertz self-routing switching node," Electronics Letters, Vol. 23, No. 25, pp. 1359-1360, 1987

25 Prucnal, P.R., Santoro, M.A., and Sehgal, S.K. (1986). "Ultrafast all-optical synchronous multiple access fiber networks," *IEEE J. Select. Areas Communic.*, SAC-4 #9, 1484-1493.

26 Prucnal, P.R., Santoro, M.A., Sehgal, S.K., and Kaminow, I.P. (1986). "TDMA fiber optic network with optical processing," *Elec. Lett.*, 22 #23, 1218-1219

27 Prucnal, P.R:, Blumenthal, D.J. and Santoro, M.A. (1987). "A 12.5 Gbps fiber-optic network using all-optical processing," *Elec. Lett.*, 23 #12, 629-630

28 P.A. Perrier and P.R. Prucnal, "Optical self-routing of 12.5 Gbit/s time-division multiplexed data," Conference on Lasers and Electro-Optics (Anaheim, CA, 25-29 April 1988), Technical Digest Series Vol. 7, Paper TUK2, pp. 74-75

29 P.A. Perrier and P.R. Prucnal, "Self-clocked optical control of a self-routed photonic switch," IEEE Journal of Lightwave Technology, Vol. LT-7, No. 6, pp. 983-989, 1989

30 K.K. Goel, P.A. Perrier, P.R. Prucnal, M.A. Milbrodt, E. Desurvire, and B. Tell, "Optical self-routing through a tree-structured space-division photonic switch using pulse-interval binary encoding," 1989 Annual OSA Meeting (Orlando, FL, 15-20 October, 1989), Technical Digest, paper MY4; "Demonstration of packet switching through an integrated-optic tree switch using photo-conductive logic gates," Elec. Lett. Vol. 26, No. 5, pp.287-288, 1990

31 E. Desurvire, B. Tell, I.P. Kaminow, G.J. Qua, K.F. Brown-Goebeler, B.I. Miller and U. Koren, "High-contrast in InGaAs:Fe photoconductive optical "AND" gate for time division demultiplexing," Elec. Lett., Vol. 24, No. 7, pp. 396-397, 1988

32 C. R. Menyuk, "Stability of solitons in birefringent optical fibers. II. Arbitrary amplitudes," J. Opt. Soc. Am. B, Vol. 5, No. 2, 392-402, February 1988

33 M.N. Islam, "Ultrafast all-optical logic gates based on soliton trapping in fibers," Opt. Lett., Vol. 14, No. 22, 1257-1259, November 15, 1989

34 M.N. Islam, C.E. Soccolich, and D.A.B. Miller, "Low-energy ultrafast fiber soliton logic gates," Opt. Lett., Vol. 15, No. 16, 909-911, August 15, 1990

35 M. N. Islam, C.-J. Chen, and C.E. Soccolich, "All-optical time-domain chirp switches," Opt. Lett., Vol. 16 , No. 7, 484-486, April 1, 1991

Figure Captions

Fig. 1 Routing using an electronic overlay network; packet header is stripped. photodetected and sent to a self-routing electronic switch that is topologically identical to the photonic switch; routing information in the header determines the state of each electronic switching element and, in turn, the state of the homologous optical switching element

Fig. 2 Illustration of optical fixed directory routing with switches A through E and stations 1 through 7; based on the destination address of the incoming packet, a decision is made at each switch as to which outgoing link the data should follow

Fig. 3 Block diagram of a 2x2 photonic switching node with optically-processed packet synchronization and fixed-directory routing; at each input, part of the signal is diverted to the optical packet-synchronization processor, which measures the time delay between the framing pulse and the local optical clock of period T_p, and tunes the optical delay, aligning the packet with the clock; part of the synchronized signal is diverted to the optical routing control processor, which identifies the packet length and address and determines the state of the switch; address field read by address recognition processor; packet-length field read by packet-length recognition processor; routing look-up table determines bar or cross state from address; state control signals compared in state-conflict resolution processor; gating pulse generator sets state of switch for duration of longer packet

Fig. 4 Optical field (address, packet-length or framing pulse) recognition processor; received signal r(t) is optically correlated with each address signature $s_i(t)$, $i=F_{(m-1)}+1,\ldots,F_{(m-1)}+F_m$; output is a vector $\{r(t)\cdot s_{F(m-1)+1}(t),\ldots,r(t)\cdot s_{F(m-1)+Fm}(t)\}$, where the entry $r(t)\cdot s_i(t)=1$ if $r(t)=s_i(t)$, and $r(t)\cdot s_i(t)=0$ otherwise; the position of the unit entry indicates the signature of the input optical field

Fig. 5 (a) Packet structure with time-orthogonal field signatures; time frame with slots $1,\ldots,F_1$ corresponds to address field; time frame with slots $F_1+1,\ldots,F_1+F_2$ corresponds to packet-length field; ith time signature corresponds to a "1" in the kth slot and "0's" in all other slots

(b) Generation of ith time signature $s_i(t)=u(t-i\tau)-u(t-(i+1)\tau)$ from optical clock delayed by $i\tau$

Fig. 6 Optical look-up table; input is a vector from address-recognition processor; toggle switches are closed (open) for the set of elements in athe vector representing addresses that should correspond to the bar (cross) state; toggles can be fixed connections or programmable

Fig. 7 State conflict resolution processor using deflection ("hot-potato") routing; conflicts are resolved by correctly routing one packet and misrouting the other; bar state has high priority, whereas cross state and unrecognized addresses have low priority

Fig. 8 Tunable optical delay producing M possible delays; feed-forward structure consists of $\log_2 M$ delay stages and an output stage; the kth stage consists of a 2x2 optical switch with a connection to the next stage at one output, and a delay $T_p/2^k$ in excess of the connectsion delay at the other output; to delay the input pulse to the jth

slot, successive bits in the binary representation of the number j is used to control each stage

Fig. 9 Simplified optical packet synchronization scheme consisting of $\log_2 r(L+1)$ stages; at each stage the packet is directed to both the 1x2 photonic switch and the framing pulse recognizer; at the jth stage, the framing pulse recognizer uses an optical matched filter with F=2 to determine whether the framing pulse is in the first (or second) half of the interval $0 \leq t \leq T_p/2^{j-1}$,and the gating pulse generator sets the 1x2 switch in the cross (or bar) state for a time duration T_p; the upper (or lower) output of the switch enters a reference delay (or a delay $T_p/2^j$); the two delays are summed and enter the next stage; after the final stage the framing pulse is aligned with the local clock with an accuracy of τ/r

Fig. 10 Banyan architecture using four-port photonic switching elements; 1xN binary tree (cross-hatched boxes and heavy lines) is constructed from N-1 2x2 switches arranged in $\log_2 N$ stages; each successive stage examines one bit in the binary-represented packet destination address, starting with the most significant bit; bit 0 (1) corresponds to upper (lower) output; path from input 110 to output 001 is shown; NxN banyan requires $(N/2)\log_2 N$ stages; internalblocking occurs when two packets are destined for the same output of the same crosspoint; two packets originating at input ports 000 and 011 with addresses 111 and 110, respectively, conflict in the second crosspoint of the second stage

Fig. 11 Self-routing 8x8 lattice architecture. If T<I then select the bar state; if T>I then select the cross state. All packets are self-routed from input to output without internal blocking. Inset: A simple time-domain approach to implementing the address comparator optically. The synchronization pulses have been removed from the set of time-orthogonal optical signatures, and the packets are separated by a guard band of duration equal to the packet header. The address pulse that arrives at one of the two switch inputs first represents the smaller address. If the address at the upper input arrives first, it triggers control input T to generate a gating pulse which sets the switch in the bar state, and the packet passes to the upper output. If the address at the lower input arrives first, it triggers control input I to inhibit the gating pulse generator, which sets the switch in the cross state, and the packet passes to the upper output. The duration of the gating or inhibit pulse is equal to a packet length, and the guard band following the packet insures that the gating pulse does not interfere with the next packet.

Fig. 12 Self-routing 2D lattice using smart pixels, formed by folding a 64x64 1D lattice in segments of K=8. Each plane is formed by taking one column of the 1D lattice, dividing it into K=8 segments, and folding it accordion-style to form an 8x8 planar array. The 1D column is laid out in a serpentine fashion on the surface of the plane as shown by the numbered inputs. Each rectangle in the planar array represents a self-routing 2x2 switching element with inputs on the front surface of the plane and outputs on the back surface. These 2x2 switching are functionally equivalent to the 2X2 switches with trigger and inhibit gating controls shown in Fig. 11. Each 2x2 switching element may contain more than one smart pixel. The simple interconnection field of the 2D lattice (all connections are simply straight ahead) provides for easy alignment and stacking of the smart pixel arry planes. For collimated input beams, no additional optics are required to provide the interconnections. Self-routing can be accomplished by simple time-domain encoding, such that the first packet to arrive triggers a gating or inhibit signal. In

this way, no additional logic is required on the 2x2 switching element to perform routing.

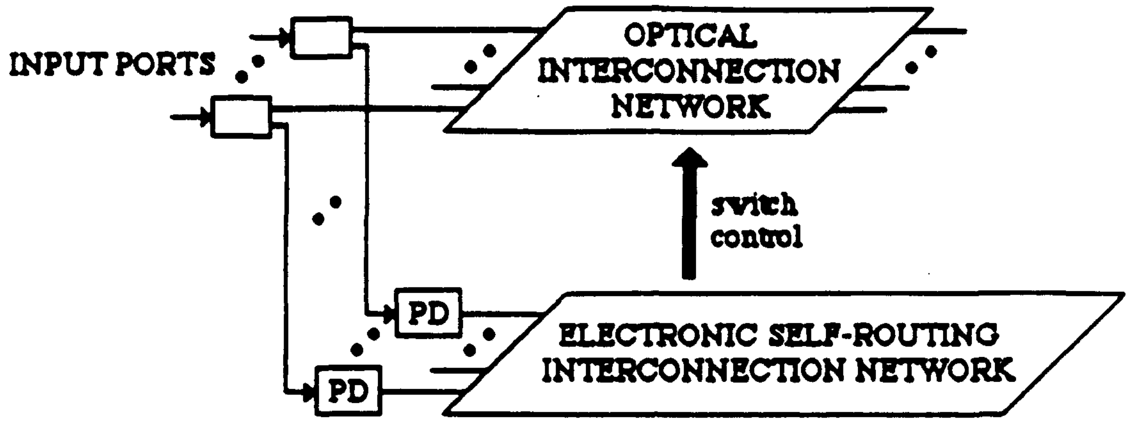

INPUT PORTS
OPTICAL INTERCONNECTION NETWORK
switch control
PD
PD
ELECTRONIC SELF-ROUTING INTERCONNECTION NETWORK

Figure 1

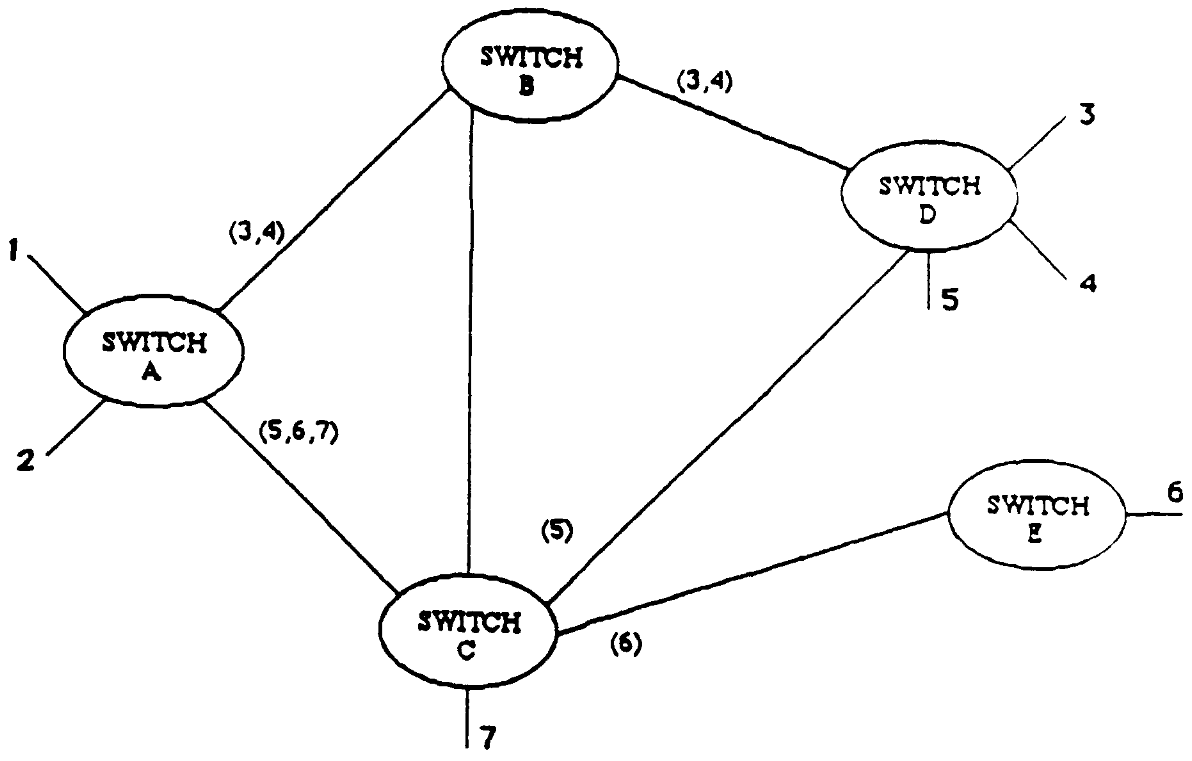
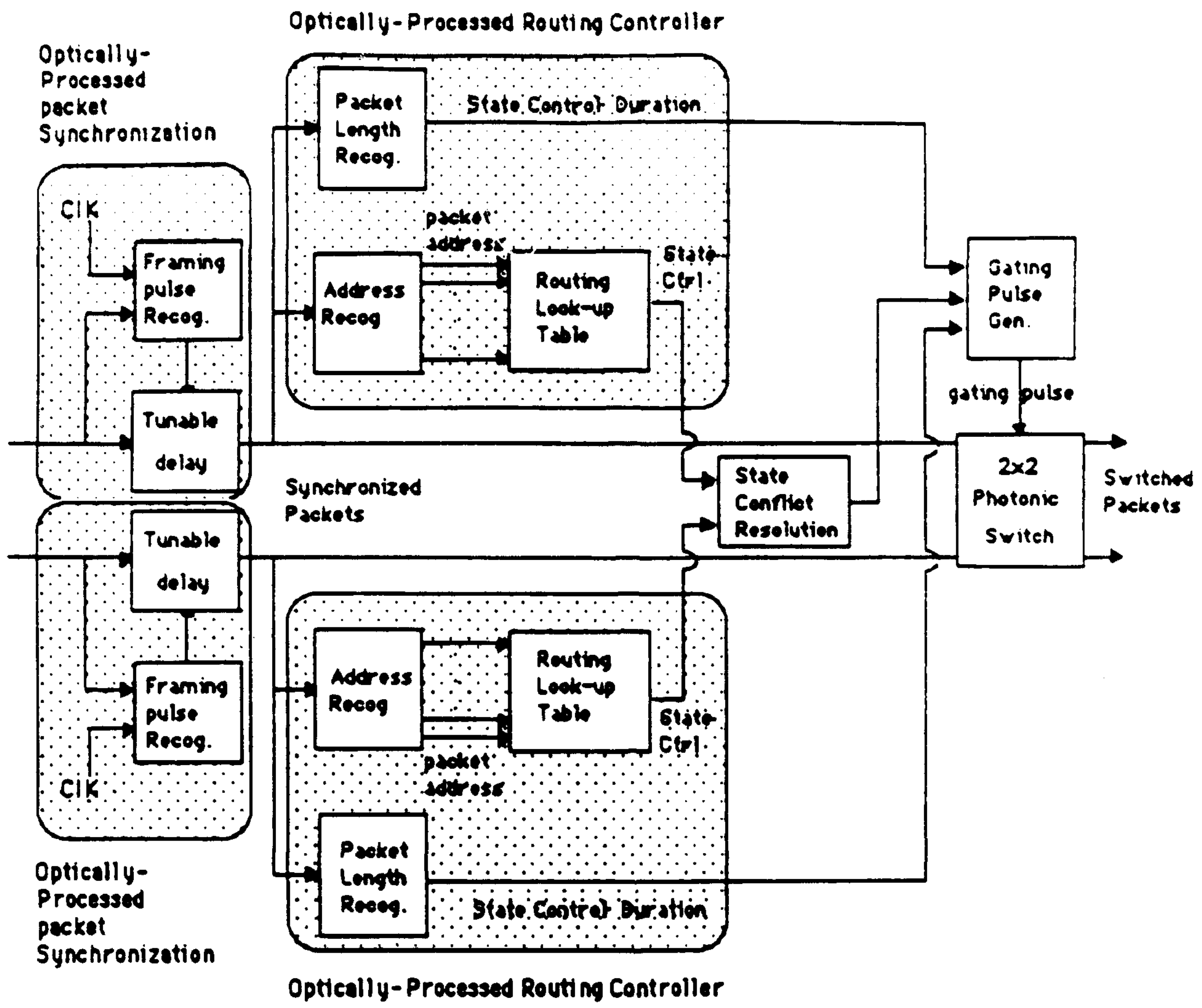

SWITCH B
SWITCH A
SWITCH C
SWITCH D
SWITCH E
(3,4)
(3,4)
(5,6,7)
(5)
(6)
1
2
3
4
5
6
7

Figure 2


Optically-Processed Routing Controller
Optically-Processed packet Synchronization
Packet Length Recog.
State Control Duration
Clk
Framing pulse Recog.
packet address
Address Recog
Routing Look-up Table
State Ctrl
Gating Pulse Gen.
gating pulse
Tunable delay
State Conflict Resolution
2x2 Photonic Switch
Switched Packets
Synchronized Packets
Tunable delay
Address Recog
Routing Look-up Table
Framing pulse Recog.
State Ctrl
packet address
Clk
Packet Length Recog.
State Control Duration
Optically-Processed packet Synchronization
Optically-Processed Routing Controller

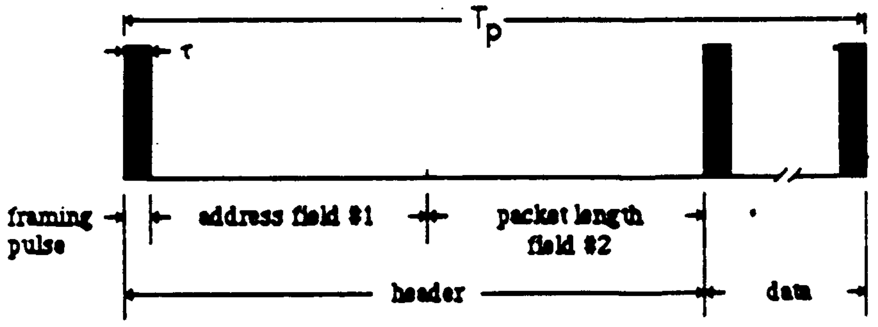

T_p
τ
framing pulse
address field #1
packet length field #2
header
data

Figure 3

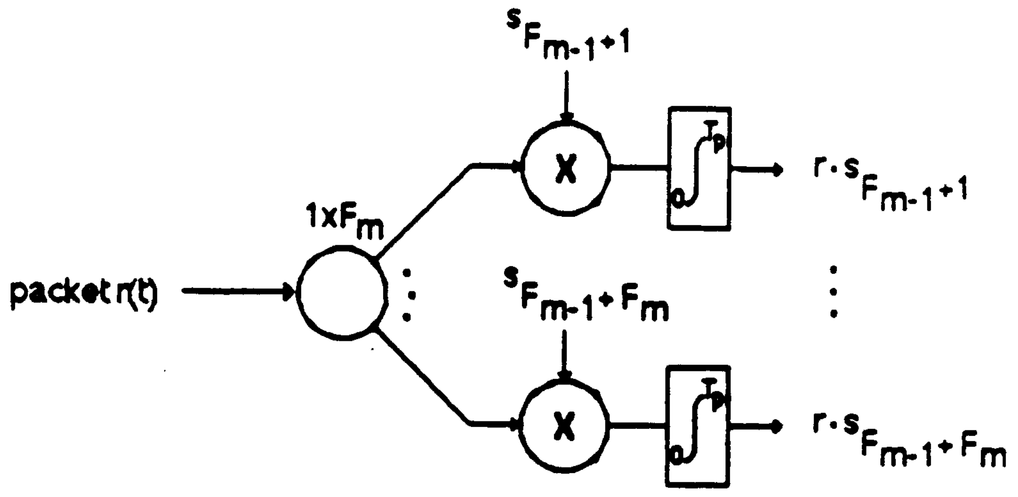
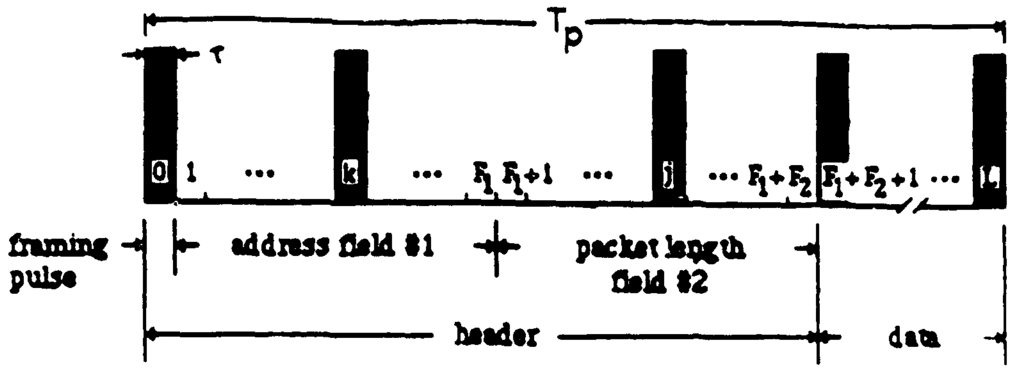

$s_{F_{m-1}+1}$
$1 \times F_m$
packet $r(t)$
X
0
T_p
$r \cdot s_{F_{m-1}+1}$
$s_{F_{m-1}+F_m}$
X
$r \cdot s_{F_{m-1}+F_m}$

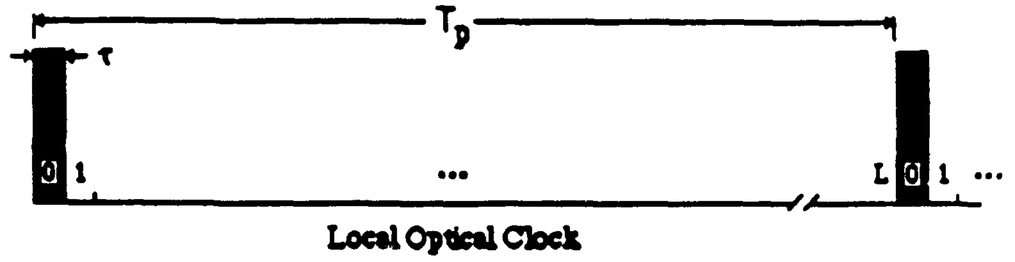
Figure 4

(a)

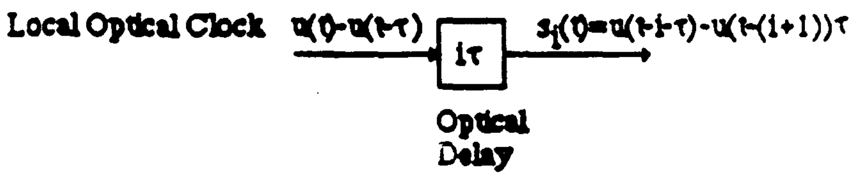


T_p
τ
0
1
k
F_1
F_1+1
j
F_1+F_2
F_1+F_2+1
L
framing pulse
address field #1
packet length field #2
header
data

(b)



T_p
τ
0
1
L
0
1
Local Optical Clock



Local Optical Clock
$u(t)-u(t-\tau)$
$i\tau$
$s_i(t)=u(t-i\tau)-u(t-(i+1))\tau$
Optical Delay

Figure 5

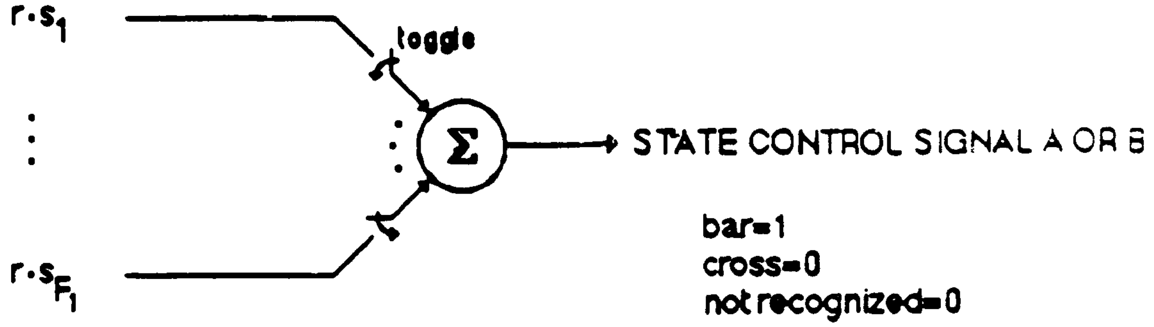
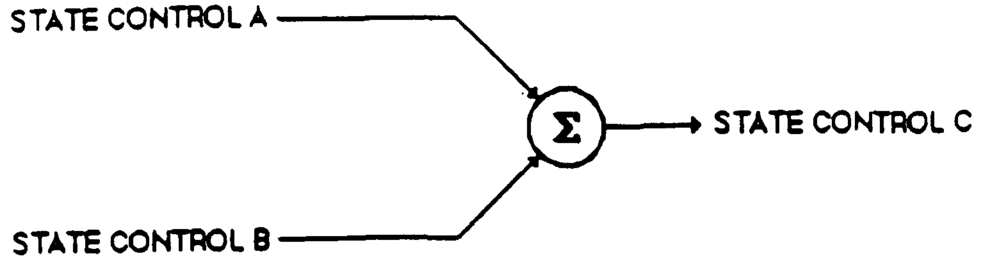

$r \cdot s_1$
toggle
$r \cdot s_{F_1}$
Σ
STATE CONTROL SIGNAL A OR B
bar=1
cross=0
not recognized=0

Figure 6



A	B	C	DECISION
1	1	>1	NO CONFLICT
0	1	1	MISROUTE A
1	0	1	MISROUTE B
0	0	0	NO CONFLICT

Figure 7

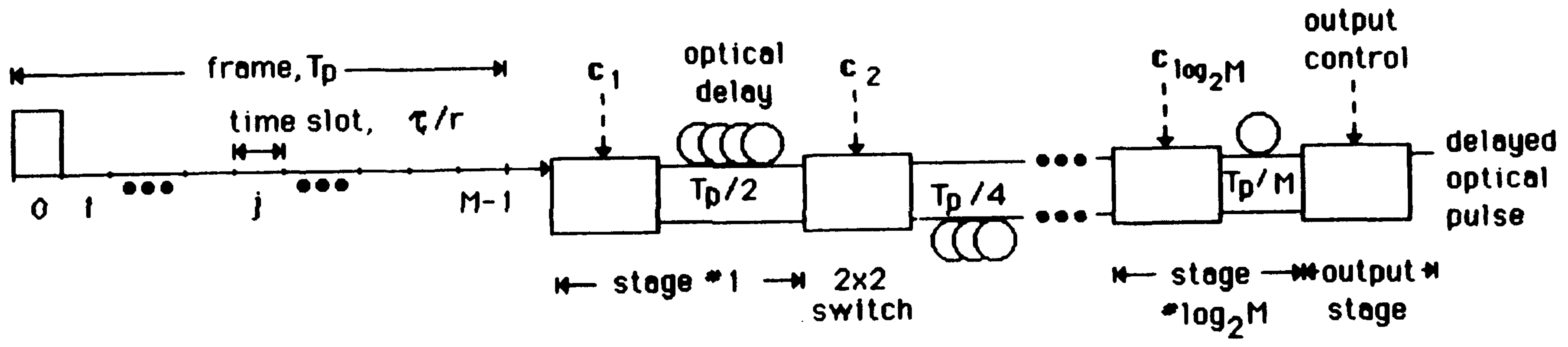

frame, T_p
time slot, τ/r
0
1
j
M-1
c_1
optical delay
$T_p/2$
c_2
$T_p/4$
2x2 switch
stage #1
$c_{\log_2 M}$
T_p/M
stage $\#\log_2 M$
output control
output stage
delayed optical pulse

Figure 8

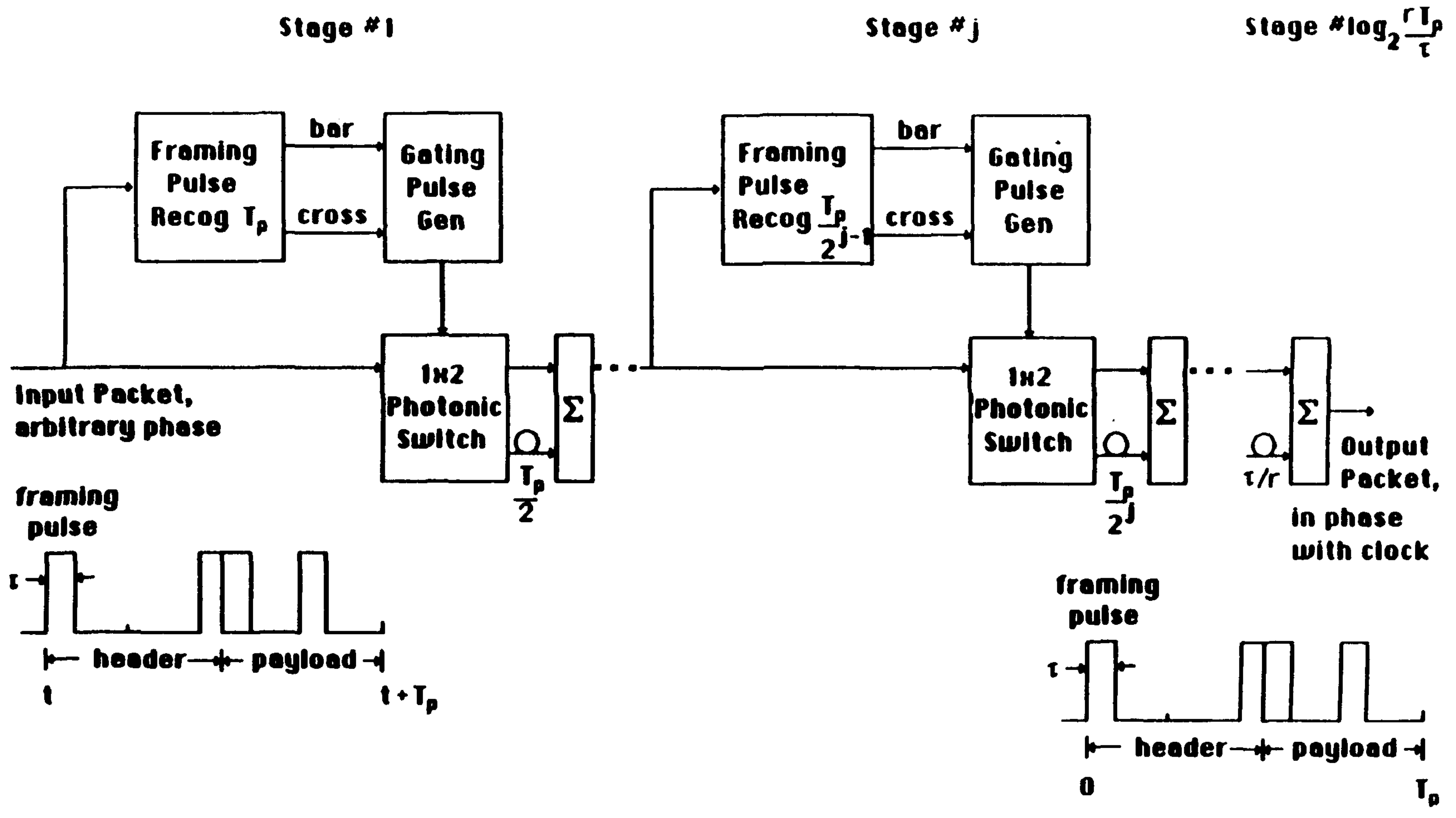

Stage #1
Stage #j
Stage $\#\log_2 \frac{rT_p}{\tau}$
Framing Pulse Recog T_p
bar
cross
Gating Pulse Gen
1x2 Photonic Switch
Σ
$\frac{T_p}{2}$
Input Packet, arbitrary phase
framing pulse
τ
header
payload
t
$t + T_p$
Framing Pulse Recog $\frac{T_p}{2^{j-1}}$
bar
cross
Gating Pulse Gen
1x2 Photonic Switch
Σ
$\frac{T_p}{2^j}$
τ/r
Σ
Output Packet, in phase with clock
framing pulse
τ
header
payload
0
T_p

Figure 9

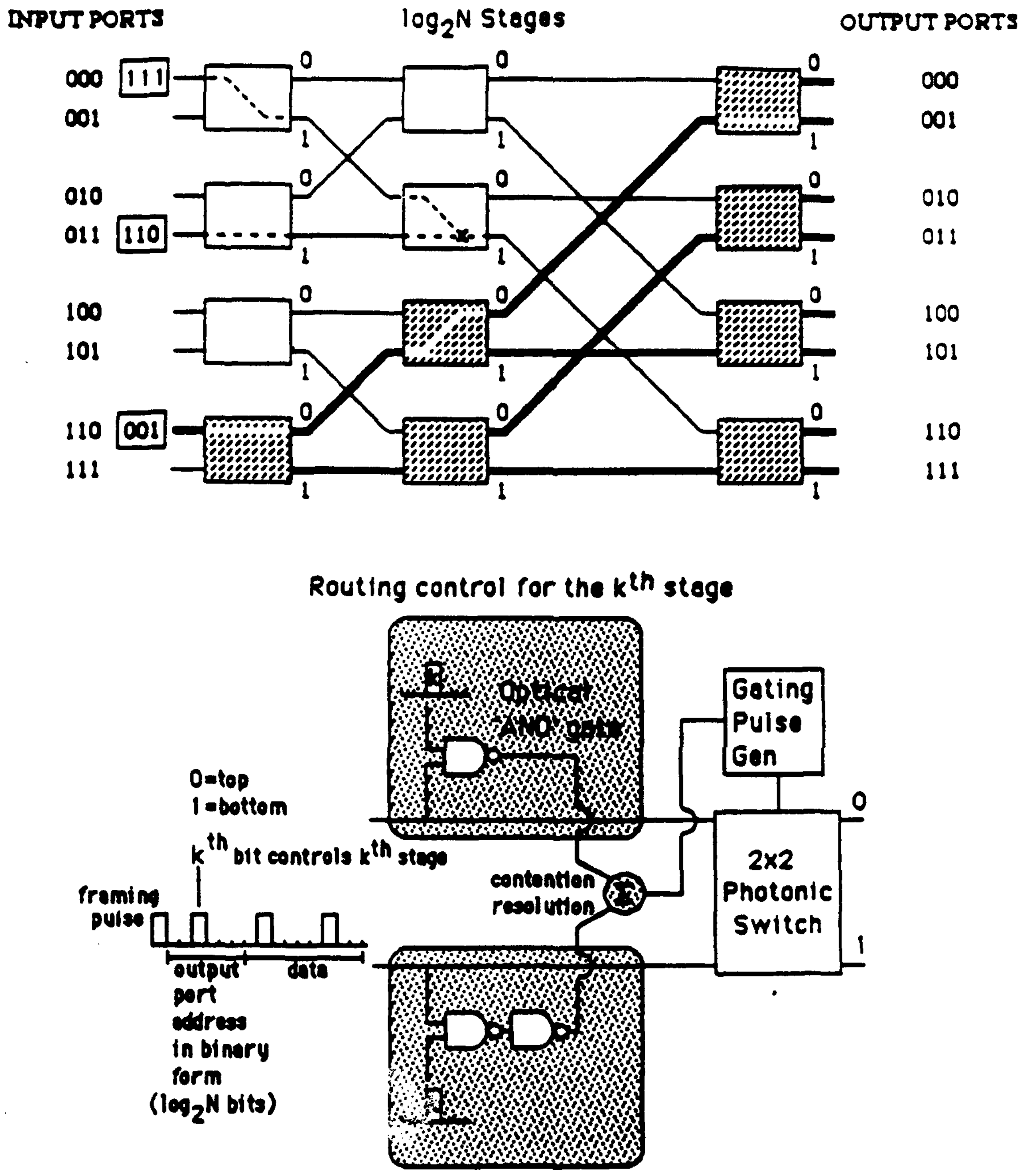
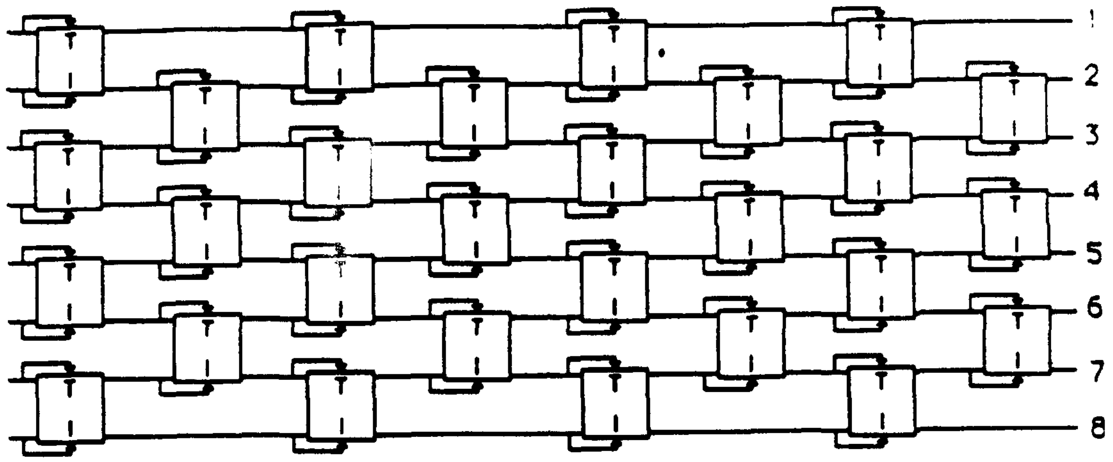

INPUT PORTS
log$_2$N Stages
OUTPUT PORTS
000
001
010
011
100
101
110
111
111
110
001
0
1
Routing control for the kth stage
Optical "AND" gate
Gating Pulse Gen
2x2 Photonic Switch
0=top
1=bottom
kth bit controls kth stage
framing pulse
contention resolution
output port address in binary form (log$_2$N bits)
data

Figure 10



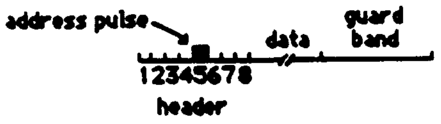


Figure 11

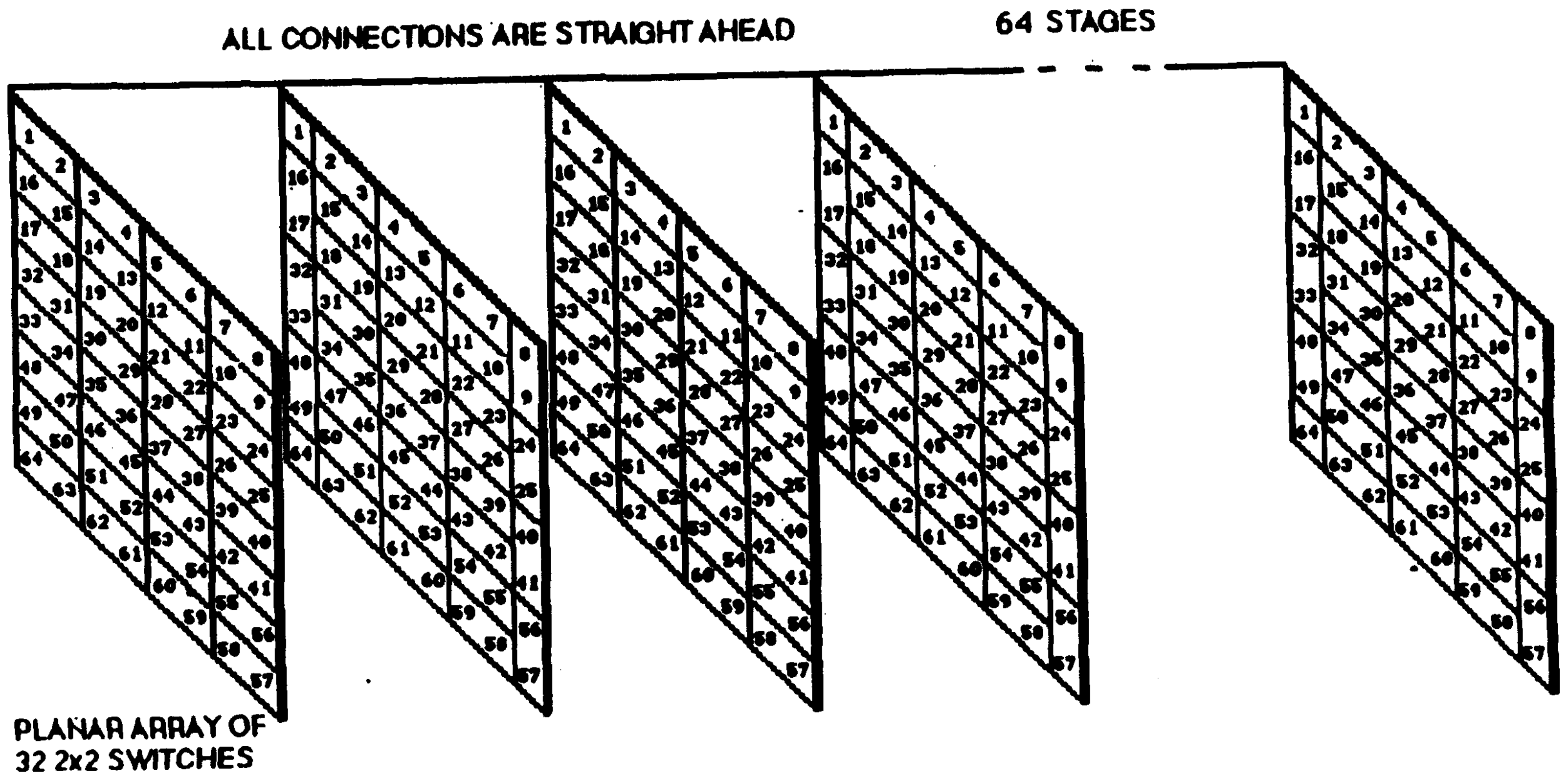

ALL CONNECTIONS ARE STRAIGHT AHEAD
64 STAGES
PLANAR ARRAY OF
32 2x2 SWITCHES

Figure 12